15. Permissions, SUID & SGID

🐧 Linux User Permissions & File Security

1. What are the Three User-Based Permission Groups in Linux?

Linux file permissions are divided into **three groups**:

- **User (u):** The owner of the file or directory.
- **Group (g):** The group assigned to the file. Multiple users can belong to this group.
- **Others (o):** Everyone else on the system who is not the owner or in the group.

Each group has permissions that control **read (r), write (w), and execute (x)** access.

2. What Are the Linux Commands `chmod`, `sudo`, `su`, `chown`, and `chgrp` Used For?

- `chmod`: Changes the file or directory **permissions** (read, write, execute).
    - Syntax examples:
        - Numeric: `chmod 755 file` (owner=rwx, group=rx, others=rx)
        - Symbolic: `chmod u+x file` (adds execute for user)
- `sudo`: Runs a command with **superuser (root) privileges** temporarily, controlled by the sudoers configuration.
    - Example: `sudo apt update` runs `apt update` as root.
- `su`: Switch user. By default, switches to **root user** shell.
    - `su -` opens a root login shell.
    - `su username` switches to another user's account.
- `chown`: Changes **ownership** of a file/directory — user and optionally group.
    - Example: `chown user file` changes owner.
    - Example: `chown user:group file` changes owner and group.
- `chgrp`: Changes **group ownership** of a file/directory.
    - Example: `chgrp developers file` sets group to developers.

3. What is the Purpose of `setuid` and `setgid` in Linux File Permissions, and How Do You Use Them?

- **`setuid` (Set User ID):**
    - When set on an executable file, this allows users to run the file **with the permissions of the file owner**, usually root.
    - Used for programs requiring elevated privileges temporarily (e.g., `passwd` command).
    - Symbol: `s` appears in user's execute bit position (`rws`).
- **`setgid` (Set Group ID):**
    - When set on a file, it runs with the **group permissions of the file**.
    - When set on a directory, files created inside inherit the **directory's group** instead of the user's default group.
    - Symbol: `s` appears in group execute bit position (`rws`).
- **Setting with `chmod`:**
    - `chmod u+s file` sets setuid.
    - `chmod g+s file_or_dir` sets setgid.

---

4. What is the Difference Between `chown` and `chgrp` Commands?

- **`chown`** changes **both user ownership and optionally group ownership** of files or directories.
- **`chgrp`** changes **only the group ownership** of files or directories.
- Use `chown` when you want to change file owner; use `chgrp` if you only want to change the group.

---

5. What Are Some Best Practices for Managing File Permissions on Linux?

- Follow the **principle of least privilege:** Only grant users the minimum permissions needed.
- Avoid giving **write permissions** to others (`o+w`) on sensitive files.
- Use **groups** to manage permissions for teams or departments.
- Regularly audit permissions on sensitive directories (e.g., `/etc`, `/var`).
- Avoid unnecessary use of **setuid/setgid** binaries to reduce risk.
- Use **ACLs (Access Control Lists)** for finer-grained permissions if necessary.
- Always secure **configuration files and SSH keys** with proper ownership and permissions.

---

6. How Can You Audit File Permissions Changes on Your System?

- Use **auditd (Linux Auditing System)** to track changes to file permissions.
- Configure audit rules, for example:

    ```
    auditctl -w /etc/passwd -p wa -k passwd_changes
    ```

    - Watches `/etc/passwd` for write (w) and attribute (a) changes.
- Review logs with:

```
ausearch -k passwd_changes
```

- Alternatively, use `inotify` tools for real-time monitoring.
- Regularly check permissions using automated scripts or configuration management tools.

---

7. What is Umask in Linux?

- **Umask (User Mask)** defines the **default permission bits to remove** when a new file or directory is created.
- It works as a **permission filter**, subtracting permissions from the system defaults (usually `666` for files and `777` for directories).
- Example:
  - Default umask: `022`
  - File created with default permissions `666 - 022 = 644` (rw-r--r--)
  - Directory created with `777 - 022 = 755` (rwxr-xr-x)
- Set or view current umask with the `umask` command.
- Important for ensuring newly created files are not world-writable by default.
